NPS55GV71091A

United States Naval Postgraduate School

SYSTEM SERVICE OUTPUT,

WITH APPLICATION TO MULTIPROGRAMMING

by

Donald P. Gaver

September 1971

Approved for public release; distribution unlimited.

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral A. S. Goodfellow, USN
Superintendent

M. U. Clauser
Provost

ABSTRACT:

The stochastic properties of the output of a multiprogramming computer system are studied by means of a simple cyclic queueing model. It is shown that output is asymptotically normally distributed. The parameters are determined by considering a cumulative stochastic process that depends upon busy period properties; the latter may be recursively determined. Numerical examples are provided.

Prepared by:

SERVICE SYSTEM OUTPUT, WITH APPLICATION TO MULTIPROGRAMMING

Donald P. Gaver*

Carnegie-Mellon University

1. Introduction

It is often mathematically convenient and useful to represent the behavior of a computer system, or part thereof, as a single server queueing process. This is appropriate even when several servers are present, as in multiprogramming situations involving cyclic queues; see Gaver [2], and Lewis and Shedler [3]. Then the server singled out for particular attention usually possesses "general" (non-exponential) service times, while the others enjoy simple Markov-convenient properties. Assuming this structure it is often possible to compute such system characteristics as waiting time properties and server idleness probability, where the latter depend upon server processing rates and the number of programs (customers) allowed to be present in the system simultaneously.

This paper is devoted to studying the distribution of the output of the server in such a process. By this we mean the following. Beginning at some moment t' the total number of service completions, during $(t',t' + t)$ is observed. Denote this number of $Z(t',t' + t)$, or by

---

Research conducted under a grant from the IBM Corporation and with the partial support of the Office of Naval Research.

---

*Currently Visiting Professor of Operations Research, Naval Postgraduate School, Monterey, California.

$Z(t)$ if the process has stationary increments. We term Z the <u>output</u> of our server, and seek to characterize its behavior. It will be shown in the sequel that in interesting cases Z is approximately normally distributed as t becomes large. Furthermore, in cyclic models, e.g., for multiprogramming, a simple continuity argument shows that the outputs of both servers enjoy the same limiting normal distribution. The methods employed make possible a comparison of various multiprogramming situations. Some limited numerical illustrations are presented.

2. Inputs and Outputs

In this section we record a simple observation upon which much of the later development rests.

(a) The M/G/1 Service System. Here λ denotes the Poisson arrival rate, and S is a generic service time. Assume $E[S^2] < \infty$. Let $N(t)$ denote the number of customers in the system at time t, and let the <u>input</u>, $A(t)$, be the total number of arrivals to have occurred in $(0,t)$. Then $Z(t)$, the total <u>output</u> in $(0,t)$, is defined by the continuity relation

$$N(0) + A(t) = Z(t) + N(t) \tag{2.1}$$

Now suppose $\rho = \lambda E[S] < 1$. Then, for large t, $N(t)$ is finite while $A(t)$ becomes large, and hence Z is asymptotically similar to $A(t)$. Putting this formally, write (2.1) as

$$\left[\frac{A(t) - \lambda t}{\sqrt{\lambda t}}\right] - \left[\frac{Z(t) - \lambda t}{\sqrt{\lambda t}}\right] = \frac{N(t) - N(0)}{\sqrt{\lambda t}} \tag{2.2}$$

Now when $\rho < 1$ and $E[S^2] < \infty$ it is well-known that

$$\lim_{t\to\infty} E[N(t)|N(0)] = E[N(\infty)] < \infty \tag{2.3}$$

and by Tchebychev's inequality,

$$P\{\left|\frac{N(t) - N(0)}{\sqrt{\lambda t}}\right| > \varepsilon\} < \frac{E[N(t)] + N(0)}{\varepsilon\sqrt{\lambda t}} \to 0 \tag{2.4}$$

for any $\varepsilon > 0$ as $t \to \infty$. Thus it follows that the left-hand side of (2.2) approaches zero in probability, and hence since the distribution

of $\frac{A(t) - \lambda t}{\sqrt{\lambda t}}$ converges to the $N(0,1)$ law, so does that of $\frac{Z(t) - \lambda t}{\sqrt{\lambda t}}$. This result will hold true for many types of queueing systems, e.g., for the GI/G/1 as well as for various multiple server configurations.

(b) The Cyclic System. Of particular interest in the multiprogramming computer system studies context is the cyclic arrangement depicted in Fig. 1.

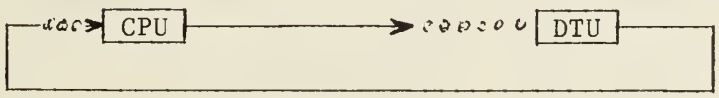

Figure 1

In the most rudimentary model a fixed finite number, J, of tasks or programs is present in the system at any time. A program is processed at the Central Processing Unit (CPU) until an interruption occurs ("page interrupt" in certain types of machines) for lack of information. At this moment the program enters the Data Transfer Unit (DTU) stage, where it awaits and eventually receives the required information and is then returned to the CPU stage. In the mean time the CPU may be busy processing another program, and is therefore kept busy. Programs that are completed at the CPU stage are assumed to leave the system and be instantaneously replaced. Models of this type have been considered by various authors, cf., Gaver [2], Lewis and Shedler [3], and Shedler [4].

Although only a limited amount of actual data analysis has been carried out, it is apparently roughly appropriate to assume that the service times of programs at the CPU are independently and exponentially distributed. Service times at the DTU are of non-exponential (more nearly constant) character.

We shall, as a consequence, make such assumptions; it is noted that this represents a reversal of the assumptions made in Gaver [2].

In order to discuss the outputs of the servers, denote by $C(t)$ the number of programs that complete the CPU stage (number of CPU service completions) in $(0,t)$, while $D(t)$ refers to the corresponding quantity for the DTU. In the present model the programs remain the same and there are no actual departures from the system. Later, we put in a departure mechanism. Next, $N_C(t)$ and $N_D(t)$ represent the number of programs at the CPU and DTU stages respectively. Then again the continuity relation (where C takes the place of Z) states that

$$N_C(0) + D(t) = C(t) + N_C(t) \tag{2.5}$$

Hence if there exist norming constants μ and σ such that $[D(t) - \mu t](\sigma\sqrt{t})^{-1}$ has a limiting normal distribution, then, by the same argument as that outlined in connection with the M/G/1-system, $[C(t) - \mu t](\sigma\sqrt{t})^{-1}$ approaches the same normal distribution. That is, the output distributions of CPU and DTU are asymptotically identical. But for the present model it is evident that $D(t)$ is actually a <u>cumulative process</u> in the sense of Smith, cf. Cox [1], or Smith [5]. Hence asymptotic normality follows, e.g., from Smith's development ([5], pp. 262-263]).

We shall, as a consequence, make such assumptions; it is noted that this represents a reversal of the assumptions made in Gaver [2].

In order to discuss the outputs of the servers, denote by $C(t)$ the number of programs that complete the CPU stage (number of CPU service completions) in $(0,t)$, while $D(t)$ refers to the corresponding quantity for the DTU. In the present model the programs remain the same and there are no actual departures from the system. Later, we put in a departure mechanism. Next, $N_C(t)$ and $N_D(t)$ represent the number of programs at the CPU and DTU stages respectively. Then again the continuity relation (where C takes the place of Z) states that

$$N_C(0) + D(t) = C(t) + N_C(t) \tag{2.5}$$

Hence if there exist norming constants μ and σ such that $[D(t) - \mu t](\sigma\sqrt{t})^{-1}$ has a limiting normal distribution, then, by the same argument as that outlined in connection with the M/G/1-system, $[C(t) - \mu t](\sigma\sqrt{t})^{-1}$ approaches the same normal distribution. That is, the output distributions of CPU and DTU are asymptotically identical. But for the present model it is evident that $D(t)$ is actually a <u>cumulative process</u> in the sense of Smith, cf. Cox [1], or Smith [5]. Hence asymptotic normality follows, e.g., from Smith's development ([5], pp. 262-263]).

3. <u>The Cyclic System, Its Busy Periods, and a Cumulative Process.</u>

The process associated with the cyclic system of Figure 1 may conveniently be viewed as a succession of busy and idle periods for the DTU. Let there be J programs circulating, and consider a moment t' such that $N_D(t'-0) = 0$ but $N_D(t') = 1$. That is, the system is idle prior to t', becoming busy with the service of one customer at t'. A busy period for the DTU is defined by

$$\tau_1(J) = \inf\{t \geq 0 | N_D(t+t') = 0\} \tag{3.1}$$

The subscript indicates that one program is present at the start of the busy period. Following each busy period is an idle period, during which all J programs are queued behind the CPU. By the Markov property of the CPU service times, a generic idle period of duration I is exponentially distributed with mean λ^{-1}. Successive idle periods and subsequent busy periods are independently and identically distributed random variables. Put

$$X^{(n)} = I^{(n)} + \tau_1^{(n)}(J) \quad \text{for} \quad n = 1,2,\ldots\ ; \tag{3.2}$$

$\{X^{(n)}\}$ represents the times between the successive regeneration points at which the DTU becomes idle. In terms of the $\{X^{(n)}$, $n = 1,2,\ldots\}$ sequence, which is one of independently and identically distributed random variables, one can speak of a renewal counting process, $R(t)$, where

$$\begin{aligned} &R(t) = 0 \quad \text{iff} \quad X^{(1)} > t \\ &\ldots \\ &R(t) = j \quad \text{iff} \quad \sum_{n=1}^{j} X^{(n)} \leq t \quad \text{and} \quad \sum_{n=1}^{j+1} X^{(n)} > t \end{aligned} \tag{3.3}$$

for $j = 2,3,\ldots$.

Let the output, or number of service completions, during a (the $n^{\underline{th}}$) busy period be

$$\beta_1^{(n)}(J) = D(\tau_1^{(n)}(J)) \quad \text{for} \quad n = 1,2,3,\ldots. \tag{3.4}$$

Then the output process $D(t)$ is cumulative in the sense of Smith [5], with

$$D(t) \approx \sum_{n=1}^{R(t)} \beta_1^{(n)}(J) \tag{3.5}$$

as $t \to \infty$ (the approximation consists in neglecting outputs during part of an X-cycle; these are negligible for large t). A central limit theorem for such processes, cf. Cox [1], enables one to show that $D(t)$, appropriately normalized, is approximately normally distributed for large t, and to find the parameters of the limiting distribution (asymptotic mean and variance) explicitly in terms of the CPU service rate, λ, and the distribution of service times at the DTU. To be specific, it may be shown that as $t \to \infty$

$$E[D(t)] \sim t \frac{E[\beta_1(J)]}{E[X]} \equiv t\,\mu \tag{3.6}$$

and

$$\begin{aligned} \operatorname{Var}[D(t)] \sim t \Bigg\{ & \frac{\operatorname{Var}[\beta_1(J)]}{E[X]} + \frac{\operatorname{Var}[X](E[\beta_1(J)])^2}{(E[X])^3} \\ & - 2\,\frac{\operatorname{Cov}[\beta_1(J),X]E[\beta_1(J)]}{(E[X])^2} \Bigg\} \end{aligned} \tag{3.7}$$

$$\equiv \sigma^2\, t$$

and that $\dfrac{D(t) - t\mu}{\sigma\sqrt{t}}$ tends to the $N(0,1)$ law as t becomes large. Explicit evaluation of the parameters μ and σ^2 is discussed

in the next section. Finally the development of Section 2 then implies that $\frac{C(t) - t\mu}{\sigma\sqrt{t}}$ also has the limiting $N(0,1)$ distribution.

4. Cumulative Process Parameters

Recursive evaluation of busy period properties as J increases was discussed in Gaver [2]. Let S be the DTU service time, with distribution $U(x)$ and Laplace-Stieltjes transform

$$u(s) = \int_{0-}^{\infty} e^{-sx} \, U\{dx\}. \tag{4.1}$$

Then consider the following cases.

Representation.

(A) $J = 1$. Here clearly

$$\tau_1(1) = S \qquad\qquad \beta_1(1) = 1. \tag{4.2}$$

(B) $J = 2$. Condition on S to find that

	Probability $e^{-\lambda S}$	$1 - e^{-\lambda S}$
$\tau_1(2)$	$S(=\tau_1(1))$	$S + \tau_1'(2)$
$\beta_1(2)$	1	$1 + \beta_1'(2)$

Figure 2.

Use of the symbol $'$ means that, for example, $\tau_1'(2)$ has the $\tau_1(2)$ distribution but is independent of events leading up to the initial service completion. To explain further, consider the situation just following the initial service completion of the busy period. Either (i) no CPU output occurred during S, an event of probability $e^{-\lambda S}$, in which case the busy period is of duration S with one output, or (ii) exactly one CPU output occurred, an event of probability $1 - e^{-\lambda S}$,

in which case the initial situation was reproduced, with one program at the CPU and one at the DTU, but with an initial component of busy period duration, S, and one initial output.

(C) Arbitrary J. Again condition on S. Define $\tau_i(J)$ and $\beta_i(J)$, $(i = 1,2,\ldots,J)$ to be respectively the first passage time from i to $i - 1$ and the output therein, given that a service is just commencing at the moment $N_D = i$.

For the present setup we then have

	Probability: $e^{-\lambda S}$	$\lambda S e^{-\lambda S}$	$\ldots$	$\frac{(\lambda S)^j e^{-\lambda S}}{j!}$	$\ldots$	$1 - e^{-\lambda S}\sum_{j=0}^{J-2}\frac{(\lambda S)^j}{j!}$
$\tau_1(J)$	S	$S + \tau_1'(J)$		$S + \sum_{i=1}^{j} \tau_i'(J)$		$S + \sum_{i=1}^{J-1} \tau_i'(J)$
$\beta_1(J)$	1	$1 + \beta_1'(J)$		$1 + \sum_{i=1}^{j} \beta_i'(J)$		$1 + \sum_{i=1}^{J-1} \beta_i'(J)$

Figure 3.

Now a little reflection shows that $\tau_i(J)$ has the same distribution as $\tau_1(J-i+1)$, and similarly that $\beta_i(J)$ has the same distribution as $\beta_1(J-i+1)$. This fact enables us to successively compute the various expectations required to evaluate (3.6) and (3.7). We now illustrate.

Expectations.

(A') J = 1. Directly,

$$E[\tau_1(1)] = E[S], \qquad \mathrm{Var}[\tau_1(1)] = \mathrm{Var}[S].$$

$$E[\beta_1(1)] = 1 \quad , \qquad \mathrm{Var}[\beta_1(1)] = 0 \tag{4.3}$$

$$\mathrm{Cov}[\tau_1(1),\beta_1(1)] = 0$$

(B') $J = 2$. Conditional on S,

$$E[\tau_1(2)|S] = S + (1-e^{\lambda S})E[\tau_1(2)]. \tag{4.4}$$

Consequently after removal of the condition on S and use of (4.1).

$$E[\tau_1(2)] = \frac{E[S]}{E[e^{-\lambda S}]} = \frac{E[S]}{u[\lambda]} \tag{4.5}$$

Likewise,

$$E[\beta_1(2)] = \frac{1}{E[e^{-\lambda S}]} = \frac{1}{u[\lambda]} \tag{4.6}$$

Squaring column by column in Fig. 2 delivers second moments. For example,

$$\begin{aligned} E[\tau_1^2(2)|S] &= S^2 e^{-\lambda S} + E[(S + \tau_1'(2))^2](1 - e^{-\lambda S}) \\ &= S^2 + \{2S\, E[\tau_1(2)] + E[\tau_1^2(2)]\}(1 - e^{-\lambda S}). \end{aligned} \tag{4.7}$$

So, upon removal of the condition on S,

$$E[\tau_1^2(2)] = \frac{E[S^2] + 2E[S(1 - e^{-\lambda S})]E[\tau_1(2)]}{E[e^{-\lambda S}]} \; ; \tag{4.8}$$

the value of (4.5) is introduced to evaluate the latter expression. Next the variance is computed by subtracting off the square of (4.5). In analogous fashion

$$E[\beta_1^2(2)] = \frac{1 + 2\, E[(1 - e^{-\lambda S})]E[\beta_1(2)]}{E[e^{-\lambda S}]} \; ; \tag{4.9}$$

insertion of (4.6) and subtraction of its square yields the variance. The covariance is obtained from the expectation

$$E[\beta_1(2)\tau_1(2)] = \frac{E[S] + E[S(1-e^{-\lambda S})]E[\beta_1(2)] + E[1-e^{-\lambda S}]E[\tau_1(2)]}{E[e^{-\lambda S}]} \qquad (4.10)$$

by subtraction of the product of (4.5) and (4.6). These expressions can be evaluated in terms of the transform (4.1) and its derivatives, and thus there is natural impetus to employ some explicitly transformable density, e.g., the gamma or hyperexponential, to represent DTU service times.

Examination of Fig. 3 makes it clear that the busy period moments for any J can be expressed in terms of the corresponding moments for smaller J-values. This step can perhaps be best carried out numerically, for neat closed-form exact expressions will not occur.

The busy period moments obtained by the procedure described may be employed to evaluate the output parameters (3.6) and (3.7). Some numerical illustrations are given in the following section.

5. Numerical Examples

The effect of assuming various parameter values in our multi-programming model can be investigated numerically by putting the results of the previous section to work. Some rather limited examples appear in the following table.

			CPU					
		λ:	0.20	0.5	1.0	1.5	2.0	5.0
DTU								
Constant, E[S]=1;	J=1;	μ	0.17	0.33	0.50	0.60	0.67	0.83
		σ^2	0.12	0.15	0.13	0.10	0.07	0.02
	J=2;	μ	0.20	0.45	0.73	0.87	0.94	0.98
		σ^2	0.18	0.31	0.23	0.12	0.05	0.01
Exponential, E[S]=1;	J=1;	μ	0.17	0.33	0.50	0.60	0.67	0.83
		σ^2	0.17	0.19	0.26	0.33	0.35	0.52
	J=2;	μ	0.19	0.43	0.67	0.79	0.86	0.92
		σ^2	0.17	0.29	0.37	0.47	0.56	0.81

Figure 4.

Notice that when $\lambda > \left(E[S]\right)^{-1} = 1$, in which case the DTU stage acts as bottleneck, the move from $J = 1$ to $J = 2$ has dramatic effects. Although the output rate can never exceed unity, improvements of at least ten percent occur. The addition of further programs $(J > 2)$ is apparently justified only if considerable overhead activity is present; this feature is not included in the present model. It is of interest to compare the numerical values of Figure 4 to those obtained by Shedler [4]. Clearly λ[CPU utilization] $= \mu$, and a reference to the

appropriate entries in Table 1 of [4] provides numerical confirmation.

Examination of the variance of output is of some interest. If λ is relatively small (CPU the bottleneck) it appears that

(a) for $J = 1$, $\mathrm{Var}[C(t) | S \text{ exponential}] > \mathrm{Var}[C(t) | S \text{ constant}]$

but

(b) for $J = 2$, $\mathrm{Var}[C(t) | S \text{ exponential}] < \mathrm{Var}[C(t) | S \text{ constant}]$

By way of explanation, one sees that when $J = 2$ busy periods are more likely to involve more than one DTU service when S is constant than when S is exponential. Of course, if $\lambda > (E[S])^{-1}$ the DTU becomes the bottleneck. As anticipated in this situation the output behaves like a renewal process with inter-event times distributed according to S. Consequently when S is constant, $\sigma^2 = t^{-1} \mathrm{Var}[C(t)]$ dwindles to zero as λ increases, reflecting the fact that outputs through the DTU bottleneck are regular. Of course, the regularity is even greater when $J = 2$ than when $J = 1$. If, on the other hand, S is exponential the variance gradually approaches that of the DTU bottleneck, namely unity.

It may be guessed from the numbers of the last table that when $\lambda > 1$ the assumption of exponential S provides an <u>underestimate</u> of output rate μ, and an <u>overestimate</u> of σ^2, provided S is more regular--of smaller variance--than the exponential. Another estimate of σ^2, useful when the DTU rate is smaller than that of the CPU, is obtained by simply assuming that the DTU is never idle, and thus

$$\sigma^2 \approx \frac{\mathrm{Var}[S]}{(E[S])^3}$$

a familiar renewal theory result that will become increasingly accurate for larger and larger J. Although we do not explore such approximations further at this point it seems evident that for increasingly complex systems--those in which there are considerations of overhead, non-exponential distributions, and in which $J > 2$--the only practical route to understanding is through approximations and bounds. If simulations are undertaken it is useful to have some idea of the variance of $C(t)$ so that run lengths may be established. Approximate variances are often adequate for such purposes.

6. Program Termination and Output

The previous development takes no account of the fact that individual programs actually terminate. In order to introduce this effect into the model, we can assume that each time a program leaves the CPU stage one of two events occurs: (i) the program terminates or is completed, or (ii) the program experiences an honest page fault and must go to the DTU stage. Suppose that choice of event (i) or (ii) is governed by a Bernoulli trials process so that with probability p the program terminates, and with probability $q = 1 - p$ the program continues to the DTU stage. In order to allow use of the previous analysis we shall assume that in case event (i) occurs a new program is immediately introduced into the system at the DTU stage; the first pass through this stage may well represent I/O activity on behalf of this newest program.

Let $M(t)$ represent individual program output over time t. Now given $C(t)$, $M(t)$ is conditionally binomial, with

$$E[M(t)|C(t)] = pC(t) \tag{6.1}$$

so

$$E[M(t)] = pE[C(t)] \sim p\mu t \equiv \mu_M t \tag{6.2}$$

as $t \to \infty$. Furthermore,

$$\begin{aligned} Var[M(t)] &= pqE[C(t)] + p^2 Var[C(t)] \\ &\sim pq\mu t + p^2\sigma^2 t \equiv \sigma_M^2 t \end{aligned} \tag{6.3}$$

and since $M(t)$ is easily seen to be a cumulative process the previously quoted theorem shows that the actual output of completed programs is approximately normal as $t \to \infty$.

REFERENCES

[1] Cox, D. R., *Renewal Theory*, Methuen Monograph, 1962.

[2] Gaver, D. P., Probability models for multiprogramming computer systems, *Journal of the ACM*, 14(3), pp. 423-438, 1967.

[3] Lewis, P. A. W. and Shedler, G. S., A cyclic queue model of system overhead in multiprogrammed computer systems; Research Report RC-2813, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, 5 March 1970.

[4] Shedler, G. S., A cyclic queue model of a paging machine; Research Report RC-2814, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, 25 March 1970.

[5] Smith, W. L., Renewal theory and its ramifications; *J. Royal Statistical Soc.*, B, pp. 243-301, 1958.

INITIAL DISTRIBUTION LIST

	No. Copies
Defense Documentation Center (DDC) Cameron Station Alexandria, Virginia 22314	20
Library Naval Postgraduate School Monterey, California 93940	2
Dean of Research Administration Naval Postgraduate School Monterey, California 93940	1
Library Department of Operations Research and Administrative Sciences Naval Postgraduate School Monterey, California 93940	3
Professor D. R. Cox Department of Mathematics Imperial College London, England	1
Professor B. Epstein Israel Institute of Technology Technion City Haifa, Israel	1
Professor P. A. P. Moran Department of Statistics Australian National University P. O. Box 4 Canberra, Australia 2600	1
Professor Daryl Daley Department of Statistics Australian National University P. O. Box 4 Canberra, Australia 2600	1
Marvin Denicoff Office of Naval Research Arlington, Virginia 22217	1
Dr. Thomas Varley Office of Naval Research Arlington, Virginia 22217	1

Dr. Bruce McDonald | 1
Office of Naval Research
Arlington, Virginia 22217

Professor Walter Smith | 1
Statistics Department
University of North Carolina
Chapel Hill, North Carolina

Dr. Glenn Bacon | 1
Systems Department
IBM Research
Monterey and Cottle Roads
San Jose, California

Professor Leonard Kleinrock | 1
Department of Computer Science
University of California
Los Angeles, California

Dr. Bill Mitchell | 1
Department of Management Sciences
California State College
Hayward, California 94542

Dr. Donald P. Gaver | 10
Department of Operations Research
and Administrative Sciences
Naval Postgraduate School
Monterey, California 93940

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

Naval Postgraduate School
Monterey, California

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

3. REPORT TITLE

System Service Output, with Application to Multiprogramming.

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Technical Report

5. AUTHOR(S) *(First name, middle initial, last name)*

Donald P. Gaver

6. REPORT DATE

September 1971

7a. TOTAL NO. OF PAGES

7b. NO. OF REFS

5

8a. CONTRACT OR GRANT NO.

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

13. ABSTRACT

The stochastic properties of the output of a multiprogramming computer system are studied by means of a simple cyclic queueing model. It is shown that output is asymptotically normally distributed. The parameters are determined by considering a cumulative stochastic process that depends upon busy period properties; the latter may be recursively determined. Numerical examples are provided.

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Computers						
Queues						
Multiprogramming						
Systems						